REMARKS

Applicants have studied the Office Action dated October 17, 2007, and have made amendments to the claims. It is submitted that the application, as amended, is in condition for allowance. By virtue of this amendment, claims 1, 2, 4-10, and 13-24 are pending. Claims 3, 11, and 12 have been canceled without prejudice or disclaimer. Claims 1, 9, and 17 have been amended. and new claims 22-24 have been added. Reconsideration and allowance of the pending claims in view of the above amendments and the following remarks are respectfully requested.

Claims 1-21 were rejected under 35 U.S.C. § 102(b) as being anticipated by Hutsch et al. (U.S. Patent Application Publication No. 2001/0034771). This rejection is respectfully traversed.

The present invention is directed to an efficient and easy-to-implement method for managing configuration data. One embodiment of the present invention provides a method for managing configuration data. According to the method, configuration values are stored in a hierarchical tree having multiple nodes, a defined structure, and defined data types for the stored configuration values. The multiple nodes include at least one inner node and at least one child node that is associated with the inner node. Each node is associated with at least one of the configuration values, and each of the configuration values dictates how an application component associated with that configuration value behaves and/or interacts with other application components. Additionally, some of the nodes are only associated with a set of configuration values, while other of the nodes are associated with a combination of a set of configuration values and an identifier associated with at least one application component. An application component is registered directly with at least one of the nodes of the tree, based on a query received from the application component. The application component is directly notified when a configuration value associated with the at least one node of the tree is modified, based on an addition or change in at least one configuration value that matches the query.

For example, in the exemplary embodiment disclosed in the specification, configuration data, which includes configuration values, is organized into a hierarchical tree. Application components

register interest in a particular node of the tree, or a particular sub-tree. The registered interest is stored within the node. Because they registered, these components receive notifications whenever a corresponding configuration value changes. Thus, any component that is a current holder of a registration or reference will be notified when a configuration value changes. Because the configuration values are arranged hierarchically in a tree such as an XML tree, the various application components can register for callbacks or notification upon modification of any nodes in any sub-tree.

Further, the structure of the configuration data in the XML tree can be altered, for example by adding more sub-nodes to a particular node in the tree. An interested party to the changed sub-tree will receive the new configuration data. Thus, an application component can register itself as an interested party to any change in a configuration value. Also, the use of XML to represent the tree allows a sub-tree of configuration data to be easily expanded, with no change in how an application component handles those configuration values. Thus, an application component (such as a Graphical User Interface, an object, an Application Programming Interface, a plug-in, an application, or a user) has the ability to handle changing configuration data, and can handle configuration data in a hierarchical structure such as by using the extensibility of XML.

The Hutsch reference is directed toward a network portal system that allows universal and integral use of different services by arbitrary client systems. The network portal system links, via a communication network, multiple content provider systems with multiple client systems. The network portal system allows a client to get content from a provider system even if the client system and the provider system do not use the same communication protocol. In Hutsch, a web-top manager within the portal system receives a content request from a client system. The web-top manager communicates with a universal content broker system that is also in the portal system. Upon receipt of the content request from the web-top manager, the universal content broker selects a content provider system that is able to provide the requested content. The universal content broker accesses the selected content provider system, and issues a request that results in the performance of the requested action by the selected content provider system.

In the system of Hutsch, if the request was to retrieve content, the content in a raw data format is passed to the web-top manager. The web-top manager renders the requested content into a page that can be displayed by the requesting client system, and then this page is returned to the requesting client system. The universal content broker system of Hutsch can include a configuration server and a configuration proxy that is coupled to the configuration server. The configuration server includes a first DOM tree, which in turn includes user profiles and application profiles. The configuration proxy includes a second DOM tree, which in turn includes a subset of the data in the first DOM tree.

Amended claim 1 recites:

> storing a plurality of configuration values in a hierarchical tree having a plurality of nodes, a defined structure, and defined data types for the stored configuration values, wherein the plurality of nodes includes at least one inner node and at least one child node that is associated with the inner node, wherein each node is associated with at least one of the configuration values, and each of the configuration values dictates how an application component associated with that configuration value at least one of behaves and interacts with other application components, and wherein some of the nodes are only associated with a set of configuration values while other of the nodes are associated with a combination of a set of configuration values and an identifier associated with at least one application component;
>
> registering at least one application component directly with at least one of the nodes of the tree, based on at least one query received from the at least one application component; and
>
> directly notifying the at least one application component when a configuration value associated with the at least one node is modified, based on an addition or change in at least one configuration value that matches the at least one query.

Amended claims 9 and 17 contain similar recitations.

Hutsch explicitly teaches that data only exists at leaf nodes of a tree. In contrast, in embodiments of the present invention configuration values exist at all nodes, not just leaf nodes. Also, Hutsch explicitly teaches that all entries in a configuration server are stored in hierarchical form in key-value pairs. In contrast, in embodiments of the present invention some of the nodes are only associated with a set of configuration values while other nodes are associated with a

combination of a set of configuration values and an identifier associated with at least one application component.

Hutsch further teaches that listeners can register to be notified when alterations are made to configuration service elements. These listeners then notify the applications affected. Because data is only stored at leaf nodes, the listeners or components can only register an interest for data in a leaf node. In contrast, in embodiments of the present invention an application itself directly registers with a node and is directly notified whenever a configuration value associated with a root node or one of its children nodes is modified. In other words, in embodiments of the present invention the intermediary "listener" is bypassed. Furthermore, Hutsch teaches that interest registries are maintained in a separate table from the DOM tree. This table is parsed whenever a change is made to an entry in the DOM tree to determine who is to be notified of the change. In contrast, in embodiments of the present invention the application directly registers with the node itself. Stated differently, the reference to an application component that is to be notified when a change occurs is stored within the node itself.

The Examiner has taken the position that Hutsch teaches that "each node is associated with at least one of the configuration values". Applicants respectfully disagree. The portion of Hutsch cited by the Examiner merely states:

> All of the entries are stored in hierarchical form in key-value pairs. As explained more completely, below a configuration tree contains separate branches for this purpose, which can be used for different users or the various user devices. The configuration tree is described through a Document Object Model (DOM) based on XML (Extended Markup Language). Communication between configuration server 336 and the various components of network portal system 100 is provided via requests for entries or requests for modification of entries. In one embodiment, the data is exchanged using an XML-based protocol.

Nowhere does this paragraph state that "each node is associated with at least one of the configuration values". In fact, Hutsch explicitly teaches throughout the disclosure that only leaf nodes of a tree include data. For example, paragraph 329 of Hutsch states that "in this embodiment, only the leaves of DOM tree 1570 can hold data. The inner nodes are used to show hierarchical relationships."

Hutsch consistently states throughout the disclosure that only leaf nodes can hold data, and the reference never teaches or suggests that inner nodes can hold configuration data. In contrast, in embodiments of the present invention the nodes include at least one inner node and at least one child node that is associated with the inner node, and each node is associated with at least one of the configuration values. Accordingly, the claimed invention distinguishes over Hutsch for at least this reason.

The Examiner also states that Hutsch teaches that "some of the nodes are only associated with a set of configuration values while other of the nodes are associated with a combination of a set of configuration values [value] and an identifier [key] associated with at least one application component." The Examiner supports this assertion by stating that Hutsch teaches a value and a key, and that this is equivalent to the recited "configuration values" and "identifier". However, there is support whatsoever in Hutsch for equating that the "key" of Hutsch is the same as the recited "identifier associated with at least one application component". Nowhere does Hutch teach or suggest that the "key" is an "identifier associated with at least one application component". Accordingly, the claimed invention also distinguishes over Hutsch for at least this reason.

Furthermore, the claims recite that: 1) some of the nodes are only associated with a set of configuration values, **and** 2) other of the nodes are associated with a combination of a set of configuration values and an identifier associated with at least one application component. Thus, to anticipate the claims Hutsch has to teach both of these recited features. Hutsch explicitly teaches at paragraph 158 that "[a]ll of the entries are stored in hierarchical form in key-value pairs". Therefore, even if it is assumed arguendo that key-value pairs are equivalent to the recited "configuration values and an identifier associated with at least one application component", Hutsch at most only teaches that nodes are associated with a combination of a set of configuration values and an identifier associated with at least one application component. Hutsch still fails to teach or suggest that some of the nodes are only associated with a set of configuration values. Accordingly, the claimed invention also distinguishes over Hutsch for at least this reason.

The Examiner further states that Hutsch teaches "registering at least one application component with at least one of the nodes of the tree, based on at least one query [transaction] received from the at least one application component." However, paragraph 159 of Hutsch merely states:

> Alterations to the DOM tree are carried out by transactions, which include inserting new nodes or new key-value pairs describing user preferences. This also covers such things as modification of individual entries or deletion of an entire subtree as the entries are no longer needed. Configuration server 336 also contains different mechanisms for querying the status of transactions or registering different listeners that are notified when alterations are made to configuration service elements and that in turn notify the applications affected.

In contrast, in embodiments of the present invention at least one application component is registered directly with at least one of the nodes of the tree, based on at least one query received from the at least one application component.

Hutsch explicitly teaches away from this recited feature of registering an application component **directly** with one of the nodes of the tree. For example, at paragraph 136 Hutsch teaches:

> In many cases, applications and components of network portal system 100 use common data. If one application modifies the properties of content in a common file, all the components using this data should be informed of the changes made. For this reason, content objects can register various listeners to the content object on content listeners registry 340. The listeners then inform the different applications when properties have been modified.

(emphasis added). Thus, Hutsch teaches away from a system in which an application component registers directly with a node, and instead teaches that a content object registers listeners on a content listeners registry.

Hutsch further teaches that:

> Notification table 1565, sometimes called notify table 1565, contains all the notification requests made by various proxies/clients. Table 1565 is consulted after each of the following transactions addNode, updateNode, and deleteNode have finished processing to check if notifications should be issued. When the client receives the notification, it also receives the snapshot of the node with changes. In

> this embodiment, each entry in table 1565 includes a session identifier, a node path, and client application information.

(emphasis added). Thus, Hutsch teaches that a notify table is parsed to determine if a notification needs to be issued. In other words, Hutsch does not register a component directly with a node, but stores interests within a table. In contrast, embodiments of the present invention register an application component directly with a node (e.g., the reference to the component is stored directly within the node of interest). This makes the notification search more efficient than storing the notification interests in a separate table.

Furthermore, paragraph 398 of Hutsch states:

> If component 1501 wishes to be notified about any changes in a subtree of DOM tree 1570, component 1501 can register via proxy 1510 for such notifications by requesting transaction requestNotify. Whenever changes happen on server 336, server 336 analyzes notify table 1563 to check which proxy or proxies should be notified. In this embodiment, the input parameters for transaction requestNotify are VectorNodes and Clientcomponent. Parameter VectorNodes is a list of node paths for which the proxy, or client components want to receive change notifications. The node paths must be part of already opened DOM tree 1570. Parameter Clientcomponent is needed only if proxy 1510 wants to be told later by server 336 which component had made the notification request and therefore should be notified. This transaction returns a status of success or failure.

This further illustrates that Hutsch teaches away from registering an application component directly with a node. Accordingly, the claimed invention distinguishes over Hutsch.

The Examiner also states that Hutsch teaches "notifying the at least one application component [listener] when a configuration value associated with the at least one node is modified [alterations], based on an addition or change in at least one configuration value that matches the at least one query [transaction]." However, the amended claims now more clearly recite "directly notifying the at least one application component when a configuration value associated with the at least one node is modified, based on an addition or change in at least one configuration value that matches the at least one query."

Paragraph 159 of Hutsch states that "listeners that are notified when alterations are made to configuration service elements and that in turn notify the applications affected". Thus, Hutsch is not directly notifying the application that is registered with the node, but is in fact notifying a "listener". The "listener" then notifies the application. Accordingly, the claimed invention also distinguishes over Hutsch for at least these reasons as well.

Applicants believe that the differences between Hutsch and the present invention are clear in amended claims 1, 9, and 17, which set forth various embodiments of the present invention. Therefore, claims 1, 9, and 17 distinguish over the Hutsch reference, and the rejection of these claims under 35 U.S.C. § 102(b) should be withdrawn.

As discussed above, amended claims 1, 9, and 17 distinguish over the Hutsch reference, and thus, claims 2, 4-8, and 21, claims 10 and 13-16, and claims 18-20 (which depend from claims 1, 9, and 17, respectively) also distinguish over the Hutsch reference. Therefore, it is respectfully submitted that the rejection of claims 1, 2, 4-10, and 13-21 under 35 U.S.C. § 102(b) should be withdrawn.

Claims 22-24 have been added by this amendment, and are provided to further define the invention disclosed in the specification. Claims 22-24 are allowable for at least the reasons set forth above with respect to claims 1, 2, 4-10, and 13-21. Further, it is submitted that limitations recited in these new claims are not taught by Hutsch. For example, with respect to new claim 22, Hutsch teaches away from "registering the at least one application component with the at least one inner node". At paragraph 329 Hutsch teaches that "only the leaves of DOM tree 1570 can hold data. The inner nodes are used to show hierarchical relationships." Therefore, when an application is interested in being notified about changes associated with data, the application is only interested in the leaf nodes. Therefore, Hutsch does not teach or suggest "registering the at least one application component with the at least one inner node". Accordingly, this claim distinguishes over Hutsch.

With respect to new claim 23, nowhere does Hutch teach or suggest "directly notifying the at least one application component when at least one configuration value associated with at least one of the inner node and the child node that is associated with the inner node is modified, based on an addition or change in the at least one configuration value". First, Hutsch teaches that an application via its listener has to register an interest in the specific piece of data it wants to be notified about. New claim 23, on the other hand, allows an application to be notified of any configuration value change or addition that occurs under the inner node. Stated differently, an application component can be registered with the inner node, but the change can occur at a child node under the inner node. The application component is then directly notified of this change. Accordingly, this claim distinguishes over Hutsch for at least this reason.

With respect to new claim 24, Hutsch is completely silent on "at least one configuration value in the plurality of configuration values that is associated with a first application component overlaps with another configuration value in the plurality of configuration values that is associated with a second application component, and the at least one configuration value and the other configuration value are nested under a common sub-tree in the tree." Accordingly, this claim distinguishes over Hutsch.

No amendment made was related to the statutory requirements of patentability unless expressly stated herein. No amendment made was for the purpose of narrowing the scope of any claim, unless Applicants have argued herein that such amendment was made to distinguish over a particular reference or combination of references.

Applicants have examined the reference cited by the Examiner as pertinent but not relied upon. It is believed that this reference neither discloses nor makes obvious the invention recited in the present claims. In view of the foregoing, it is respectfully submitted that the application and the claims are in condition for allowance. Reexamination and reconsideration of the application, as amended, are requested.

If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is invited to call the undersigned attorney at (561) 989-9811 should the Examiner believe a telephone interview would advance the prosecution of the application.

Respectfully submitted,

Date: January 17, 2008

By: /Stephen Bongini/
Stephen Bongini
Reg. No. 40,917
Attorney for Applicants

FLEIT KAIN GIBBONS
GUTMAN BONGINI & BIANCO P.L.
One Boca Commerce Center
551 Northwest 77th Street, Suite 111
Boca Raton, Florida 33487
Telephone: (561) 989-9811
Facsimile: (561) 989-9812